A Unique4Ps-Based Approach to Teaching Introductory Software Engineering

Lili Hai[*1], Kathleen P. Greenberg[2]

Computer Information Science Department/Psychology Department, SUNY College at Old Westbury
Old Westbury, New York, USA

[*1]hail@oldwestbury.edu; [2]greenbergk@oldwestbury.edu

Received 24 June 2013; Accepted 18 August 2013; Published 19 March 2014

Abstract

In computer science, software engineering is a special area that requires disciplines from many aspects. These aspects incude something hardly taught in the college software engineering courses. The normal methodology is to let students involve into a project, in different levels, depending on the curriculum design in different programs (introductory, multi-course, master degree, etc.). It is a real challenge to design a project based course to teach decent contents in a single software engineering introductory course of a CIS undergraduate program. This paper introduces a teaching methodology which exposes students who have only one software engineering course in 4-year curriculum to a special and stimulating learning environment. The method is based on the concept of the Four Ps in software engineering: people, project, product and process. By analysing the necessary learning elements in software engineering, including core topics in each of the four Ps, the learning environment is set by using a project which consists of the carefully selected learning elements to allow students to learn software engineering actively and efficiently with limited resources and time. The method also provides a way of easing the project design for a software engineering introductory course.

Keywords

Software Engineering Education; Unified Software Development Process; Teaching Elements; Learning Environment; Project-based

Introduction

By its nature, software engineering (SE) knowledge is grounded in scientific theory. More importantly, however, it is a set of well organized engineering principles collected and summarized from the rich experience of large scale software development. Software engineering has a very special role in the field of computer science. Its discipline is not solely based on knowledge of computer science (e.g., methods of analysis and design, modeling, programming, algorithms, etc.). It also includes elements of management, and requires good communication skills, domain knowledge, and an awareness of ethical issues. Accordingly, SE educators have realized the importance of putting students into a practicing environment in order to help them acquire the full range of the knowledge and skills that comprise SE in its entirety. This has led to the development and exploration of other teaching approaches, including lecture/project, project only, and other novel methods [Aygűn 2004; Sebern 2002; Petkovic et al. 2006; Vliet 2000; Dubinsky et al. 2005; Germain 2005] as an alternative to the traditional lecture lab style. Alok Mishra et al. [2007] conducted a survey among some leading universities that offer SE programs. Their study focused on the dimensions of *interdisciplinary skills, practical experience, communication skills and continuing education*, and showed that practice courses are a major part of the SE programs in those universities. The teaching philosophy underlying these courses stems from the idea that by participating in a project, students will not only learn key SE concepts, but will also gain training in a broader range of skills that will increase their ability to handle issues in a real world context.

Although this overall approach has been shown to be an efficient and fruitful way to teach SE, differences in curricula across institutions that grant a degree in SE or design multiple SE courses have led to the development of different approaches to the design of SE project-based courses [Debbie 2009; Bagert and Mengel 2005; Blake 2003; Hilburn and Humphrey 2002]. One of the biggest design challenges concerns the issue of how to adapt the elementsof an SE practice course to use at the SE introductory level for CIS majors, where there is a need to pack both fundamental SE principles and disciplines into a short

period of time (e.g. a single semester). The purpose of this paper, therefore, is to explore the issues that arise when designing such a course, and to describe the features and outcomes of a practice course in introductory SE that we have been teaching on a consistent basis over the past eight years.

Background

One approach to the SE project-based course is to assign students real world projects sponsored by industry [Fincher et al 2001; Alzamil 2005; Ford et al. 1995]. However, there are some difficulties in applying this method [Aygűn 2004]. In most cases, it is hard to involve novice students directly in real world projects. Another difficulty is how to efficiently combine the contents of the curriculum with the real world project in a limited time.

To design an efficient SE teaching and learning environment for these types of project, it is necessary to consider the content that may be included in the curriculum, to define it as a set of specific teaching objectives (or learning outcomes), and to find a way of teaching the content to meet the educational goals. In order to do so, we must answer the following questions:

- What elements should be selected into the project-based SE course?
- How should we teach those elements through the project?
- How can we design the project, or the learning environment, for CIS majors who have **a single SE course** in their undergraduate education?
- How can we assess the effectiveness of the course in achieving its curricular goals?

During the past several years, we experimented with a method by which we tried to find some answers to the questions. The goal of the experiment was to integrate as many SE learning elements as possible while allowing novel students to easily absorb SE principles and practice fundamental SE disciplines at their level. To reach the goal, our specific challenge was to design the learning environment appropriate for an introductory SE course. Although the literature contains a report of a teaching process called UPEDU [Robillard et al. 2001] that is based on Rational Unified Process [Jacobson et al. 1999], an O-O SE process, UPEDU is designed for higher level courses in multi-course SE programs [Germain et al. 2005; Dulipovici et al. 2004; Duim et al. 2007]. Compared with UPEDU and the methods used in mentioned literatures, we will introduce and describe a project-based methodology which follows a process customized from the **Unified Process** but designed specifically for the type of undergraduate CIS program that includes ***a single SE course***. The main goal of the method is to incorporate some of the teaching elements for a high-level SE project into a low-level student project, and find a way to teach these elements effectively. The paper solely explored the issues appeared in applying the Unified Process to the project and suggested the solution to solve the problems.

To better understand the method presented in this paper, we offer the following description of the background of the course setting and the student population in the course (in our college):

1. Software Engineering is a required course in the curriculum for Computer Information Science (CIS) and Management Information Systems (MIS) majors. The major prerequisite for this course is Data Structures and Algorithms.
2. To teach a project-based course, it is important to understand the "human resource" factor. There are three types of students:
    1) Second-year students. Students should have completed Data Structures and Algorithms.
    2) As Software Engineering is not a prerequisite for Database Management in our curriculum, the students either have taken the Database Management course, or they may be taking it at the same time.
    3) Transfer students. (These students normally are not second-year students. They have learned some mid-level courses in their previous schools, but they have never learned software engineering or similar courses.)

After understanding the student population in each specific class, the project design for the class could be more in accordance with the features of the class. However, no matter what types of students there are in each class, the main idea of the teaching strategy is the same: let students learn the fundamental concepts and principles through a large project and follow Unified Process.

On the basis of eight years of experience with this methodology, we have been able to identify a set of selected teaching elements that are based on the software development concept of the 4-Ps [Jacobson et al. 1999], *People, Project, Product and Process*. In this paper, we will 1) discuss these concepts and how they were used to select the elements to be included in the project environment design; 2) describe how the SE knowledge and principles that comprise each P can be

learned in the project through cooperation between instructor and students; 3) discuss the advantages and disadvantages of incorporating the selected teaching elements in the project and describe problems and experiences with the use of the method; and 4) provide data on the degree to which the method meets its teaching objectives. It was concluded with some final thoughts and suggestions for future work.

The 4Ps and Learning Elements

The 4-Ps are four key concepts in software engineering [Jacobson et al. 1999]. Almost every topic in the field of software engineering can be categorized into one of them, which implies that no one of them should be omitted from our teaching goals, even for the introductory SE course. In 2004, ACM and IEEE sponsored the development of a guideline, Software Engineering 2004 (SE2004) [IEEE and ACM: The Joint Task Force 2004], for creating the curricula of undergraduate SE degree program courses. However, both the concepts included in the 4-Ps and the content covered in SE2004 are too broad. Following SE2004, the initial idea of creating our methodology was to find 'core' elements from the document and fold them into the 4Ps. This is because 1) students find the 4P conceptualization to be easy to understand and 2) an approach that focuses on the 4Ps provides a way to incorporate these core elements into our teaching. The step was to analyse and select the largest minimum set of the most basic and important elements from each P to design a SE project-based course. Table 1 shows the results of those two steps, including selected SE knowledge and skill teaching (or learning) elements covered in each P and the possible mapping to a project-based course.

The teaching objectives are set based on the "learning elements" in Table 1. The methods used to achieve and measure the learning outcomes are in the "()" following each element in the table.

- The methodology for designing such a course consists of the following:
- Providing a work environment that allows students to be exposed to scenarios that people may encounter in a real world project.
- Customizing a real-life project that is large enough to expose students to the many key software development disciplines that cannot be gained through working on a series of small projects.
- Selecting the project in a manner that allows the students to best utilize and further enhance the skills they already possess.

TABLE 1 LEARNING ELEMENTS

Ps	Mapping	Learning Elements/Practicing
People	The instructor and all students in the class	In professional practice: ✓ team work (See section ***Teams***) ✓ team member behavior and communication skills (See section ***Communication***. Occupies 15% in total grade) ✓ concept of stakeholders (Through stakeholders' meetings) ✓ the way of dealing with different types of people involved in a project (Teams are formed by instructor)
Project	A single project for all people in one semester	In project management: ✓ management concepts ✓ project personnel and organization (Shown by team forming and many meetings of running project) ✓ project planning (Scheduling in different levels and stages) ✓ product estimation (See section ***What Students Can Do***)
Product	One complete release of a software product	In product artifacts and related knowledge: ✓ domain knowledge (By creating domain model) ✓ software modeling concept and major UML diagrams (See section ***Modeling***) ✓ artifacts in each workflow (Implementing all artifacts in the project) ✓ software reuse (e.g. Java library and inheritance classes in the project) ✓ design patterns (The patterns used in the product software) ✓ software architecture (The subsystems decomposition and assignments) ✓ OO design and programming (Implementation process) ✓ user interfaces (in the product)
Process	A customized Unified Process	In the software development process: (See section ***Process***) ✓ concept of process ✓ product specification and requirement elicitation ✓ analysis ✓ architecture design ✓ object design (in Javadoc format) ✓ implementation (Tool: NetBeans for coding and DB design/implementation) ✓ tests (unit test and integration test) ✓ process iterations ✓ traceability

A common question about this approach is how the project is finished and how the required SE principles and knowledge are learned at the same time. The answer to this question is that the class adopts a **lecture/meeting/lab** mixed style. And, we do two tasks. One is to prepare the project as a tool to introduce SE to students; and another one is to prepare students in order for them to be able to act appropriately (learn efficiently) in the project. To fulfill these two tasks in a limited time period, our strategy is that except for the introduction of basic and key concepts, the major topics in lectures are about the technologies of creating the artifacts of the ***product***. The knowledge of the **other three Ps** is mainly condensed through various activities (meetings/labs) over the course of developing the product in the project.

About the Assessment

In the most cases, class sizes ranged from 10 to 15 students. Although the course was introductory in all cases, the classes were at varying levels due to the varying abilities of the students in each class. The class difference may cause the quality of the products different. But it affects neither the teaching elements nor the criteria of the assessments based on the teaching elements. Following the table 1, in the 4Ps project-based course, our general teaching objectives are that the students demonstrate the understanding the concepts and the ability of working properly in the doamins of the 4Ps.

The following methods or criteria were used to evaluate the outcomes:

- Quizzes (10%) and a midterm examination (35%).
- Assignments in each workflow of a process (team/individual) for creating artifacts. The final integrated one is the product (40%).
- Performances in the project such as the team member behavior, meeting attendance, communication skills, etc. (15%)
- Astudent survey at the end of the course. The questions were designed against the teaching objectives

We categorize the assessments on the basis of the teaching elements mapped to each P in this paper, and present them in each related section. If not especially indicated, the statistics are based on the data collected from 2011 to 2013 (with n=60, n is the total number of students in different projects).

People

For simulating a real project, the organizational structure of the class is as follows:

- The instructor is the supervisor of the department of a 'company' (also the senior software engineer, the organizer, the secretary and many other things).
- The whole class is divided into teams
- Each team has a team leader
- All students are "employees"

Combined with lectures, all of the following issues discussed in this section are introduced and explained to students as a model of the organization that underlies a real project.

Multi-role

The concept of the stakeholders may be introduced when "people" of the 4Ps is discussed. Students will be the stakeholders themselves and work cooperatively to each other during the project. For students to better understand the different roles of "people" in a project, a role's title should be explicitly indicated. Each student is helped to understand the title and the responsibility of the role and to know whom to contact in order to solve a specific problem in each workflow (or phase) of the software development process.

In project Hotel Reservation System, for example, one student maectively in different workflows. The point here is to make sure that the students understand the responsibilities of each role and the position of the role in the project.

Teams

Teamwork does not simply mean breaking students into teams and assigning work for the teams to finish. In a company's department, a team member is an employee first. Besides personal interest in the job and willingness to take on responsibility, an employee has to work under many constraints. But a student does not. This discrepancy requires the instructor to design the program with a focus on student engagement as well as on the acquisition of knowledge. Students' engagement in the project is prerequisite for being good team members. On the other hand, it is necessary to create some constraints, including the proper grades for the collective work of the team, for individual behaviors affecting the quality of that work, for individual contributions to the team, and so forth. An additional purpose of these constraints is to train young students to treat the team's interest higher than

their individual interests. Without common interests, the respect, help, sharing, and cooperation between team members, the learning goals are difficult to achieve.

Debbie Richards [2009] has made a comprehensive survey on group formation and assessment in project-based courses. It discussed the con and pro of the student teams which are "self-selected vs. organized by the lecture". For a semester-long SE project, considering maximizing the outcomes of the project, the teams are organized by the instructor in our method and use a type of the heterogeneous style which allow instructor to shape the nature of the group and avoid problems caused by self-selected teams [McConnell 2006]. This method allows students and their teams to mutually benefit by ensuring that each team works efficiently and contributes in an essentially equal way to the development of different parts of the product. The rules behind making the teams are as follows:

- Based on the results of a techniques and self-evaluation survey at the beginning of the semester.
- Each team should consist of students who are strong in different areas.
- The race is one of the factors to consider.
- If the instructor knows students well, the students' personality types [Layman et al. 2006] may be considered.
- Be easy for each team to select the team leader.

Even though the project size is based on the class size, the size of teams is 3 or 4 for all of projects. As Richards [2009] indicated "It appears that teams of three learn to work harder and better, perhaps suffering less from the problems associated with communication overhead and dividing up and recombining a task ……".

Once teams are formed in the second week of the semester, major activities of the project are accomplished in teams. The teams have the freedom to make many decisions such as the communication style, in-stage schedules, etc. In this environment, students can always locate their position as team members and quickly gain a sense of belonging to the team.

Report Scheme

The reporting system is an efficient way to train students to take responsibility as a team member. For understanding the progress of the project, reports of current status of the work progress or encountered problems from teams to the 'supervisor' should be sent through the team leader periodically, similar to the reporting scheme in real companies. This way, the team leader is the first one to understand the current task situation in the team, then the supervisor. It gives a team leader room to deal with some minor problems. A summarized report written by the supervisor is posted back to the whole 'department'. Anything that blocks the progress of any group's work is transparent to all. A student understands that a delay or a vital problem in his/her part will affect not only his/her own progress (thus the grades), but also the team's progress (thus maybe the team's grades) and the success of the whole project.

Team Leaders

Team leaders play an important role in a project; hence, the criteria upon which this selection is made are of crucial importance. Team leaders should have a good understanding of the lecture material, be able to identify and raise issues, organize meetings, and technically lead the teamwork under the guidance of the instructor. One point in software engineering is that a good manager may not necessarily be a good developer. However, in a low-level SE student project, a team leader should first be a good programmer (i.e., have performed well in programming courses) because we do not have other criteria to judge the capabilities of a student at the time that the teams are formed.

In this method, we have tried both assigning leaders by the instructor and self-nominating and voted on by the team. For the latter, the criteria for selection are announced beforehand.By our experience, the projects with assigned leaders tend to have better quality.One benefit of using the self-nomination method is that it reflects a student's willingness to take on the responsibility of team leadership. In general, team leader has a little more workload than team member. In turn, the team leader benefits by obtaining real team management training and by earning points that will be applied toward their final grade.

Communication

Communication is the most basic and important skill in SE; it is also one that most students lack. In SE, there are many things that cannot be taught, and can only be acquired through training or experience. Communication is one of them.

The followings are the kinds of communication skills that can be trained in this simulated software development environment, and these may help to resolve the above problems to some degree.

- *Contact List* – A list of e-mail addresses and phone numbers of all students posting to the class
- *Talk* – Several minutes of in-class casual talk among students taking place before lectures or meetings to simulate the "hallway talk" that often takes place in companies
- *Meeting* – In-class meetings to solve technical problems or to make major decisions on project
- *Meeting Minutes* – Students writing minutes for important meetings about project issues
- *E-mails and Phone Calls* – Messages require a response within a certain time
- *Report* – Periodic progress reports from team leaders and team members
- *Presentation* –Presentingresults of an artifact by teams
- *Course Management Tool* – Angel Learning Management Suite is used to send e-mails, record all useful web links, post announcements, archive the meeting minutes, and post/ store artifacts submitted in every workflow

Team-team communication is another important part of the training in this environment. Many tasks such as identifying interfaces (boundary conditions) between subsystems require cooperation of students from different teams.

Assessment on 'People'

In this course, 15% of the student's grade is based on the performance of *team member behaviour* including

- communication skill (reading/writing, response time to reply e-mails, and reporting): 10%
- meeting attendance: 5%

By reviewing students' communication performance, the aspect of training students' communication skills in this method may still need strengthening. The average grade on communication in thelast three projects is 82%. This indicates that communication is a big issue in students' project, and that there is much room for improvement in students' level of success at meeting the criterion of being a good team member. Encouragingly, most students do tend to improve their team member behavior by the time the project enters the design workflow.

By analyzing the number of reports and the report frequency in the two projects of 2012, we have noticed that about two-third of all reports are made in the second half of the semester, with the highest frequency occurring in the design workflow. This implies that there is a learning curve for students as they get to know the function and importance of the report scheme and begin to appreciate the effectiveness of working as team members as the project progresses.

Virtually most students (83%) agreed that they felt as though they were working in a highly structured, constrained environment; 89% said they used the report scheme to deal with issues related to the project and about 89% felt the project taught them to be a good team member; 90% said that it is "completely true" that coordination between team members and teams is a key factor to the success of the project.

It was found that the frequency of the supervisor and team-leaders' meeting directly affects the usage of the report scheme. When team leaders prepare for meeting with the supervisor, they collect working status from team members. This enforces the report scheme functioning because team members have to describe the problems they encountered, have solved and are solving. The time is also for students to seek help. Among 89% of students (mentioned in above), around65% used the report scheme to solve the issues encountered in the project "every time/almost every time", the rest used it in "some level" and only 10% of total "rarely use" it.

Nonetheless, it will be necessary to continue to explore the issue of instructor, team leader and team member interaction so that the teaching method can be modified to promote the development of a more collaborative environment, and help students better understand and appreciate the value of operating within the constraints of the scheme.

Project

"Project" is defined as "the organizational element through which software development is managed" [Jacobson et al. 1999]. In this method, students have a chance to become seriously involved in project management at different levels.

What Students Can Do

The following are some aspects of project management that students may participate in the project:

1) Meetings

Through participating in different styles of meetings, students can learn meeting management and understand the difference between a meeting in a project and a regular classroom discussion. A

meeting must solve specific problems in order to let the development process proceed and reach the project goals. Thus, to have efficient meetings, especially those for solving technique problems, all teams must prepare for the meetings.

2) Estimation

Simple descriptions of more than one possible product are offered to the students at the beginning of the semester. Under the guidance of the instructor, students estimate the time frame, the possible artifacts that must be included and the resources needed, including people, existing systems, tools, and lab environment. Based on the estimates and common interests, all students vote for one product to work on. Through participating in the product estimates, students tend to be more responsible for and engaged in the project they decide to do.

3) Task scheduling

Since students have no idea precisely which activities they will do at the beginning of the project, the instructor should make a rough schedule for the whole project and announce it during the early stages of the project. Students later can participate in the planning of the sub-schedule for each workflow. As a result of making the schedule, participating in activities that follow the schedule and making necessary updates to the initial schedule, students come to understand the importance of project scheduling. Students learn concepts such as milestones, product delivery date, etc. in a natural way.

4) Risk management

Some things that cause project failure in the real world, such as cost overruns and domain environment changes, never occur in a SE class project. However, the following risks are possible: erroneous self-assessment in the screening survey, delayed interface coordination caused by mis-understanding among team members or teams, the work of one person or one team is behind schedule, and a key person withdraws from the class during the semester. Students are guided to recognize the risks during the project and learn the strategies to prevent or handle the problems.

Project management is a special field in SE that is hard for students to practice. The goal of the method is to let students understand the elements in the field and gain experience at some level. The above elements are chosen as the cut points to allow every student the opportunity to learn by doing.

Workload Distribution Models

Vertical Style – One student plays a limited role in the project by focusing on the tasks in a specific area. For example, one 'employee' could be an architect who should be responsible for the architectural work in each workflow.

Horizontal Style– Every student plays as many key roles as possible in different workflows. This will allow students to gain a wider range of knowledge and skills.

The vertical style is simple for distributing the work, and each person's responsibility is clear. The consistent role playing allows a student to understand the role's responsibility better, to learn the related knowledge in depth and to work efficiently. However, this style fits advanced capstone projects more than the project in introductory SE course because the students' training may be too narrow. Considering the expected learning outcomes of the introductory course, we adopted the horizontal style. However, some additional difficulties have to be overcome such as requiring extra communication, more work to monitor the distribution of tasks and control the project's pace.

Project Size Control (Planning and Controlling)

So far, all projects selected for experimental classes based on the method have been in the business soft-ware category:Hotel Reservations, Library Management, Student Information Management and Online Purchasing (simulated). The size of a chosen project depends on the size of the class and the "human resources" available. Once the basic product features are specified, the project scope must be decided. Our strategy is that the functions of each product are prioritized and divided into controllable components. After identifying the key functions that must be implemented, the rest of the system functions can be organized into optional service packs to balance the workload among teams. For example, in the Online Book Store system, the required functions are browsing, searching and purchasing, and the optional extended functions include login, stocking, billing, shipping, etc. Students participate in finding all possible functions at first and then identifying key use cases under supervision. After that, the key use cases are distributed to the teams and the reason for the distribution is explained. Through this activity, students learn how to make decisions about mapping

the amount of work to existing resources.

Normally, the number of functions identified and included in requirement elicitation tends to be larger than the number finally implemented. For example, the ratio of identified use cases to implemented use cases in the project during the fall of 2009 was 24:16.The problem should be solved in the analysis phase when use cases are transformed into analysis classes and the workload becomes clearer to students. The instructor needs to guide students to avoid having students drop functions at the last minute.

The risk control is another important topic in "project". Dur to the page limitation, we do not discuss it here.

Assessment on "Project"

In the survey mentioned in section 2 at the end of the course, students were asked about the degree to which they participated in key activities and what they learned from working on the class project. The data indicate a high level of participation.

Overall, near 80% students claimed that they understood and participated in the project management (in different levels). Most did so through their involvement with product estimation and selection (86%) and by attending team meetings which related to the project management issues (71%). What is particularly notable, however, is that students did not merely participate in the project, but were personally invested in the outcome. This is evidenced by a mean rating of 9.43 when students were asked to rate on a scale from 1 (*not at all*) to 10 (*a lot*) how much they cared about the success of the project.

One point here is that the teaching objectives in this P are that let students understand the possible project management work in software development. In this simulated environment, obviously, a regular team member has less chance to practice on project managementcomparing with a team leader. In some projects, we opened the supervisor and team-leaders' meeting, which is the chance to learn handling the project management related issues, to the whole class. For example, in the fall of 2012, about 30% regular team members showed up in the meetings regularly. This indicates that students understand the opportunity of learning "project" through various meetings and want to utilize the chance voluntarily.

Product

The accuracy of the artifacts created in the project is one of key criteria in evaluating the results of our teaching. Learning elements in "product" are directly reflected by what the students are supposed to accomplish in the project. Among many elements, a few are chosen for discussion in this section.

Learning Domain Knowledge

In order for students to learn the requirements of collection and elicitation, each project starts from a vague text description of the product. Each product is open-ended. To obtain the correct requirements, students have to learn the basic business domain knowledge. This strategy allows students to have better understanding of the stakeholders, the application domain, the business entities and the system they will develop. It also eases the work of building the use case model.

To shorten the time period for getting knowledge about the domain, projects in the business domain are selected in accordance with the net-generation's domain knowledge, such as online shopping (simulated), library management, etc. To overcome the difficulty of students having little domain knowledge about the system under development, students should be guided to visit the people working in the business domain. Otherwise, the instructor should prepare clear and detailed guidelines about the chosen domain.

The first project meeting is held after teams are formed and the product is chosen. This meeting simulates the activity of system engineers to capture the requirements from clients and users at the start of a project. Meeting attendees are 'users', 'clients', and 'system engineers' from each team. This meeting is an opportunity to share the domain knowledge among main stakeholders and developers (students) under the guidance of the domain expert (invited people or the instructor). The result of the meeting will be used as the detail problem statements, a base for identifying use cases.

Modeling

Teaching modeling is one of the most difficult jobs in the introductory SE course. Even though students have learned object abstraction and some algorithms to handle data structures, modeling an entire application system is completely new to them. In the method, students will learn to model the system by different UML diagrams: *use case diagram*, *statemachine diagram*, *class diagram*, *collaboration diagram*, *sequence diagram* and subsystem diagram. Students are also

required to create the *domain model diagram* of the system once the domain knowledge is initially acquired.

The difficulty of teaching modeling to students taking a lower level SE course arises because students have to grasp the following at the same time. They must:

- Understand the system to be modeled (domain knowledge may not be gained completely at the time of modeling);
- Understand the purpose of each model (which specific angle the system is viewed from and who may use this model);
- Learn the diagram used to create the model (what a node or an edge represents in the diagram); and
- Abstract the target system and use the diagram to represent it.

Most of the time, students learn concepts of the models and usage of UML diagrams through the lectures, but have trouble applying them until they practice them in their project tasks. To students, learning to abstract the system is crucial and should be a step by step process. From relatively simple models in initial stage to relatively complicated models in latter stages, our experience is that techniques and experiences we want students learn in one stage can be gained by maximally utilize the knowledge and information acquired from the previous stage(s) and more importantly, by following the natural order of the workflows in the SE process.

We use one class period to introduce modeling concept and commonly used UML diagrams at the early semester and do not introduce any diagram in detail until the time the diagram must be used in the right workflow. To the students who have no modeling experience, this way allow students to learn modeling efficiently by proceeding in an orderly way and step by step. Table 2 lists some facts found in this matter.

Identifying Use Cases

In Unified Process, the use-case model drives all subsequent workflows in the process. So, achieving the right product heavily depends on how well the use case model is created in 'Requirements'. It is crucial that students understand and practice defining the use cases. As indicated in table 2, students are still struggling to understand the application domain while they learn abstraction and modeling. More time is required to help facilitate and enhance their level of understanding in the stage of 'Requirements'.

Due to the popularity of use cases in current software engineering of the real world, every student is required to define and in charge of some use cases of the system, called the *owner* of the use cases. This strategy also helps to define the APIs of classes in design workflow because each use case owner has to guarantee that the functions needed in his/her owned use cases are not missing, even in the classes that may be designed and implemented by other people.

During the capturing of use cases, the difficulties the students have include understanding what exactly a use case is and how big it should be. Besides introducing the rules for identifying use cases, instructors should strongly suggest that students simulate scenarios to identify use cases. The conversation between two students playing roles of an 'actor' and a 'system' begins by an actor requesting a service from the system. Once the actor gets the 'value' wanted, the communication between the two parts plus the work done by the 'system' in the conversation is an instance of a use case. Experience has shown that this is a more efficient and easier way for students to scale a use case than that by making them consider its functions on the basis of a general description only. This method also makes writing the use case detail description easier.

API or MPI? (with early implementation)

In the introductory SE course, normally the students have little experience designing class APIs in a large system. In the 'Design' workflow, many students do not follow the design decisions and would arbitrarily change the interface by themselves. We call this "My Preferred Interface (MPI)".

This problem occurs when students do not have a clear concept of API and they are weak in abstract data types. Another reason that students change the design decisions at will is that they cannot clearly see the entity object control flow in the classes under design. On one hand, the instructor can use this chance to review related concepts and techniques in OO design and programming. On the other hand, according to our experience, the traditional data flow diagram of the entire system is helpful in illustrating the flow of business data processing before sequence diagrams are created. It is better to present and discuss this diagram after the initial design class list is obtained at an early stage in the design workflow. Students then have a picture of the whole system to help them

understand the context of each class within the software architecture and how the class works in the whole system.

In this method, each team identifies the design classes involved in its subsystems and class APIs. Then, all teams together confirm the accuracy of the APIs and eliminate any redundancy. After that, individual team members may implement the design classes assigned to them. Due to the dependencies among subsystems (or components), some APIs may have to be defined and even implemented before the same can be done for other APIs.

For second-year CIS students, the design in software engineering provides a perfect chance to review and enhance the concept of API in OO design and programming. Students only truly grasp the meaning of API after designing APIs by themselves in the SE project. However, as said in [Henning 2009], "… for every way to design an API correctly, there are usually dozens of ways to design it incorrectly." Under the teaching goals of this course, we can only minimize the gaps in the API design; we should not expect perfection until the complete correct set of APIs obtained in implementation and test.

Assessment on "Product"

Along the process, students (teams) are assigned the work (teamwork and/or individual work)to create key artifacts in each workflow. These assignments occupy 40% of total grades and reflect a student's performance in learning "product".

TABLE 2 SOME FACTS FOUND IN STUDENTS' LEARNING MODELING

Model & UML diagram	Knowledge based on	Difficulties	Performance (Average)
Domain model diagram ***(domain model)***	➢ Textually understanding of the problem statements ➢ From personal experience or domain expert	➢ Identify domain entities ➢ No modeling experience ➢ Tend to use nodes to represent actions	88%
Use case diagram **with detail descriptions** (use case model)	➢ Textually understanding of the problem statements ➢ Initial domain knowledge ➢ Use case scenarios	➢ Weak in modeling experience ➢ Still struggle to understand the target system ➢ Identifying of a use case (what should be included in a use case)	
Analysis classes (stereotypes) ***(Analysis Model)***	*Entity class:* ➢ Domain model ➢ OO programming ➢ Possible data collections in application *Boundary class:* ➢ Use case diagrams ➢ Daily life knowledge about user interfaces *Control class:* ➢ The functions not covered by entity and boundary classes in a use case.	The first time to abstract the system into "classes" in high level abstraction	84%
Collaboration diagrams ***(Analysis Model)***	➢ Understanding of use cases ➢ The types of Analysis classes ➢ Better understanding of domain Note: the collaboration diagram is very valuable for the new UML learners. See [Hai 2009].	➢ The first time to realize use cases by classes ➢ The functions of control classes are not very clear	
Subsystem diagrams (Architecture Model)	➢ The initial workload distribution among teams by instructor at the beginning of the project ➢ Understanding of principles of decomposition (from lectures) Note: this model is built in class meeting and confirmed for the basic components in each subsystem by each team	Analyze the subsystem boundaries for the functions requiring objects cross multi-subsystem	N/A (not an assignment)
Class diagrams (Design model)	➢ The analysis classes ➢ The collaboration diagrams ➢ The experience of modeling gained in previous workflows	➢ Understanding meaning of APIs (see section 6.4) ➢ The communications to define right APIs	87%
Sequence diagrams (in Design model for several key use cases only)	➢ The design classes ➢ The collaboration diagrams	➢ The diagrams tend to be complicate, especially the ones involve more than one entity classes	

From the survey, we know that 92% of the students agreed that the course helped them understand

(extremely well or very well) the differences between a programming assignment in a programming course and the implementation of a software *product*, 90% strongly agreed that they used the concepts and knowledge in the lectures to guide their software development activities, and 92% agreed strongly that the project made it easy to understand the importance of defining correct interfaces among subsystems in a large product. These findings show that the method does work efficiently in the context of a project-based course, and that it does achieve the objective of allowing students to learn by doing.

Table 2 shows some facts in student's learning modelling. We obtained the 'average performance' of each activity by collecting data (grades of the related assignments by the same instructor) from the projects, and performing the necessary calculations. The grades include both team work and individual work.

Process

The most efficient way to learn the software development process is by going through all workflows to finish the project. That is the basic idea of the method. Since the Unified Process featured by three software engineering methodologies: *use-case driven*, *architecture centric*, and *iterative and incremental*, we discuss the issues of applying these features in the process of thestudent projects.

Use-case Driven

After the use-case model is created, all later workflows are about realizing the use cases. The use case realization is a sequence of activities occurring in a process which transforms the requirements into the final product. As a use case owner, each student has to be responsible for guaranteeing that the use cases he/she owns will eventually be implemented and function properly. The tasks a use case owner may do include:

- *Writing detailed descriptions of the use cases.*
- *Identifying analysis classes and creating collaboration diagrams from the use cases*
- *Identifying design classes and creating sequence diagrams of the use cases*
- *Implementing assigned classes (work as class owners)*
- *Testing the use cases*

Being a use case owner, a student clearly sees how the requirements are transformed into program classes in the OO software engineering process. This scheme also helps define class APIs of quality (owners check the missing functions and take care of the data flows). Nearly even distribution of the use cases among students avoids the situation that some students work only on one type of job, say writing the document within the team, without practicing on other key works in the whole project.

Architecture Centric

System architecture is usually very vague to students when first introduced in lecture. The activities related to the architecture permeate almost all workflows. Software architecture is generally not easily designed into student projects to allow students practice in this area. In this method, related issues (what the students can learn) include:

1) Architecturally Sensitive Projects

Because identifying subsystems or components is one of the architecture training objectives, the domain of the selected product needs to be large or broad enough to allow the system to be broken into two or more subsystems. The subsystems are not necessarily big, as long as they are functionally independent. If the selected project is not architecturally sensitive, students may have little chance to solve system architecture issues in the project.

2) Architecturally Working Distribution

In our method, the teams' assignments are basically mapped to decomposed system functions (subsystems). There is a conflict here. We want to form teams early in the project; we want to form the teams based on the potential subsystems which cannot identified early. To solve this problem,the instructor has to take the responsibility to assign tasks to teams in the early stagesdepending on the functions in potential subsystems. However, the reasoning behind the assignment of workloads should be clearly explained to students, as this is an important initial study about the software architecture. The formal system decomposition should be done in the "System Design" workflow.

3) Subsystem Identification

One of the difficulties students have in identifying subsystems is that data shared by several subsystems confuse the boundary of the subsystems. In accordance with the relatively independent and internally coherent design principles of the subsystems, communications between regular subsystems are normally

accomplished by sharing data objects, not by talking directly to each other. An efficient way is to adopt data repository pattern by using database. In this way, the shared data and its management are abstracted as a general service package and accessed by all other subsystems. This requires students self-learning JDBC and creating database.

4) Subsystem Interface Identification

To identify the interfaces among subsystems, each team has to submit two lists, a list of "use" interfaces listing all interfaces used by "this" subsystem from other subsystems and a list of "provide" interfaces listing all interfaces provided by "this" subsystem. To correctly produce the lists, team-team meetings are necessary. Through this activity, students see the interfaces among subsys-tems clearly and alsolearned the layers dependency.

Iterative and Incremental

Due to the limitations of the project size, the available time, and human resources (the students cannot be treated as real experienced developers), one project cannot include many iterations. Also, the initial iteration may not include all the *core use cases*. This is because the knowledge and techniques required for implementing some complex core use cases may span several major SE disciplines that the students have not yet learned and will thus need a longer time to complete.

We require students to collect and define all functions in the 'Requirements' and then identify the key use cases. Compared with real-world projects, the first iteration in the students' project contains simpler use cases such as 'viewXXX' and 'addXXX', in which the functions are straightforward and do not involve more than one or two shared entity classes. It is desirable to start the complex use cases only when students have already gained experience and understand the system better. In some projects, all use cases were covered in requirements, analysis and design. The concept of iterations only applied on implementation and test.

Good timing is critical to the project plan. Students should participate in mini project planning in order to understand iterative and incremental development; the instructor should closely monitor students' activities to ensure that the schedules are followed.The traceability between models is one learning element that unfortunately may be easily ignored in the students' project. In the process, the forward tracing is natural. But the backward tracings are normally omitted when handling some changes. It is necessary to train students to review the changes against the related artifacts created in previous workflows and then make the decision to change.

Assessment on "Process"

Importantly, the first-hand experience students obtained by participating in the project coupled with the high level of commitment this teaching method engenders seems to have effectively facilitated the acquisition of one of the key learning outcomes – specifically, that software engineering is the *process* by which a software product is developed. In the survey, all (100%) of students agreed that they understand the software development *process*extremely well or very well after taking the course.

Through working in a complete project, students participated in all activities in all workflows. Most students (93% strongly) agreed that they understand the basic activities in each workflow for creating various artifacts through working on the project.

Evaluation: Quality of Products

In a project-based course, the quality of the final "product" somehow reflects the outcome of the teaching in some level. Here we discuss some issues in evaluating the "product" produced in the project. The quality of a "product" is affected by two major factors:

a) The overall design and control of the project (The effort of the instructor)

One may be concerned that for a student project of an introductory SE course designed for CIS majors, the learning elements described in the above sections are too much. Our experience is that as long as the learning environment is well designed and the students are fully engaged in the project, the project will be successful and the outcome will meet our teaching objectives. A study done by Helen Chen et al. [Chen et al. 2008] shows that "whether referring to specific components of an undergraduate education or to the full experience, faculty members exert a critical role in creating conditions conducive to student engagement." The success of the student project is closely related to the learning environment created by the instructor(s). This success is not only evaluated by the physical production but also by the number of SE principles taught and discipline trained in

quality.

The approach we introduced allows the instructor to design the SE learning environment based on elements collected in the 4Ps. Although the environment does give students the chance to work independently through active learning, it is still the instructor who carefully elaborates each key link of the project. Experience indicates that the students have better engagement in a meticulously designed environment than in a rough one.

The size of the project should not exceed the capability of the students in the class. An estimation of student capability can be done using the students' self-evaluation survey at the beginning of the project and can be adjusted by the students' work in the first several weeks. The real size of the project is determined when choosing the core use cases. On average, after analysis classes are assigned, besides the simple object classes, each student should not own more than two or three classes which involve complicated functions. The instructor must insist on the report system and closely monitor the progress of the project.

b) *The average level of the students in a class; that is, the students' level at the beginning of the project and their average learning capability.*

Some students with work experience (especially, related to IT) working as team leaders would strengthen the management of the project. In that case, team-team communication is more active. This makes all activities requiring team cooperation easily controlled and leads to better quality of the products. Some students have taken or were concurrently taking a Database Management course and were willing to apply what they learned in that course to the project. This affected the architecture design of the product. The products using databases usually have better quality than those using data structures and external files to manage persistent data. The quality of the final product is also related to the programming skill of the students.

One of the facets of our method is that among all the possible activities in developing an entire project, the instructor can flexibly select the proper activities depending on current human resources. Again, therefore, instead of using the quality of the final product to judge the success of the project, we would prefer to judge it by the number of activities which have been accomplished successfully.

About Data Analysis

In this paper, the most analysis given are based on the data collected from the projects of 2011 to 2013. As the number of CIS majors declined in the year of 2005 to year 2010, the average number of enrollments in each class (thus in each project) is 7-8. As the methodology discussed in this paper were applied to the classes as the experiment in the first several projects, it has being improving year by year. Compare with the data given in above each section, the data collected from 2005 to 2010 are lower. Table 3 shows some student survey results from 2005 to 2010. It is indicated that the methodology is now maturer, and the larger classes which can do larger projects give students more benefits from the method.

TABLE 3 STUDENT SURVEY FROM 2005 TO 2010

Question (samples)	Strly Agr (%)	Agr (%)	Neutral (%)	Disagr (%)
The course helped me understand the process of software development	78	22	0	0
At the end of the course, I understood the differences between a programming *assignment* in a programming course and the development of a software product	67	22	11	0
I learned principles of software engineering through lectures, and I used those principles to guide my tasks in the project	67	33	0	0
Working on the project made it easy for me to understand the basic activities in each development workflow	56	44	0	0
Working on the project made me realize that it's very important to define correct interfaces among subsystems when working on a large project	56	40	0	4
In the class, I felt like I was working in a highly structured, constrained environment in which I had to work closely with other team members, communicate with my team leader and teammates, and take responsibility for the success of the project	56	40	4	0
Working on the project taught me to be a good team member	42	44	11	2

Note: All data for "Strongly Disagree" are zero. Due to the size of the table, the column is deleted from the table.

Conclusion

Founded on the concept of the 4-Ps, *people*, *project*, *product* and *process*, the SE teaching method introduced in this paper provides a project-based learning environment for a single SE introductory course in CIS major's curriculum. Built around the key concepts and disciplines of SE, this method treats the development of a large software product as a chain consisting of a sequence of well organized activities. Each node in this chain contains some key knowledge or skills, called learning elements in this paper, each of which falls into one of the 4Ps. The instructor can select the content of the course from the above chain based on specific curricular goals, and flexibly organize the project based on the "human resources" provided by students with varying levels of prior knowledge and experience. For better evaluation and monitoring of the students' work, a task from the chain can be chosen as a formal assignment by the instructor, depending on its importance and the time required. Through a single carefully selected project and the virtual relationship between people in class, students are exposed to a simulated environment where they are trained in a wide spectrum of substantive SE disciplines including project management, professional practice, and key activities in the software development process.

To strongly engage the students so as to allow them to actively learn the SE principles and gain training in the SE discipline, the faculty members themselves must at first be knowledgeable and engaged. The project must be carefully designed and flexibly handled. The assessment of the course indicates that this teaching model both bolsters students' interest in the learning of SE and demonstrates the teaching efficiency of the method. One tough area for instructors is to keep students under the constraints of their role within the designed environment. Students tend to show much more confidence and enthusiasm towards the project once they become familiar with the novel environment.

In conclusion, it seems not only feasible, but advantageous to adopt the current method in the introductory SE course. By allowing students to gain first-hand experience with the SE process, the approach provides a strong foundation for those who will be pursuing more advanced SE courses, and helps students adopt a correct way to work on their project in the capstone course or in their future work.

Currently, efforts are made to apply the 4-Ps to teaching system maintenance by utilizing completed projects from previous classes. Compared with the projects that start from collecting requirements, the maintenance is even more difficult and more closely simulates the real world situation.

REFERENCES

Alzamil, Zakarya. "Towards an Effective Software Engineering Course Project." ICSE (2005), St. Louis, MO, 631-632.

Aygűn, Birol, "A Platform for Software Engineering Course Projects." Turk J ElecEngin, 12 (2), 2004.

Bagert, Donald. J., and Mengel, Susan. A. "Developing and Using a Web-based Project Process Throughout the Software Engineering Curriculum." Journal of Systems and Software, 74 (2) 2005, 113-120.

Blake, M. Brian, "A student-enacted Simulation Approach to Software Engineering Education", IEEE Trans. Educ., 46 (1) 2003, 124-132.

Chen, Helen. L., Lattuca, Lisa. R., and Hamilton, Eric. R. "Conceptualizing Engagement: Contributions of Faculty to Student Engagement in Engineering." Journal of Engineering Education, 97 (3) 2008, 339-353.

Dubinsky, Yael, and Hazzan, Orit."A Framework for Teaching Software Development Methods." Computer Sciences Education, Vol. 15, No. 4, 275-296, Dec. 2005

Duim, L., Andersson, J., and Sinnema, M. "Good Practices for Educational Software Engineering Projects." Proceedings of the 29th International Conference on Software Engineering, 698-707, Minneapolis, May 2007.

Dulipovici, M., and Robillard, P. N. "Cognitive aspects in a project-based course in software engineering." Proceedings of the Fifth International Conference on Information Technology Based Higher Education and Training, Istanbul, TURKEY, May2004.

Fincher, Sally A., Petre, Marian, Clark, Martyn A. C., boyle, R. D., Capon, P., Evans, G., Mander, K., and Milne, W. Computer Science Project Work: Principles and Pragmatics, Springer Verlag. 2001.

Ford, David, and Paynting, Richard."Linking Academic Theory and Industry practice with Student Interactive Projects."The Center for Quality Management Journal, 4(2) 1995: 19-32.

Germain, E., and Robillard, Pierre. N. "Engineering-based

processes and agile methodologies for software development: a comparative case study." The Journal of Systems and Software, (2005) 75, 17-27.

Hai, Lili. "The Role of Collaboration Diagramsin OO Software Engineering Student Projects." proceeding of 2009 IEEE-CS Conference on Software Engineering Education and Training, 2009.

Henning, Michi. "API Design Matters." Communications of ACM, 52 (5), May 2009.

Jacobson, I., Booch, G., and Runbaugh, J. The Unified Software Development Process, Edison-Wesley, 1999.

Layman, L., Cornwell, T., and Williams, L."Personality Types, Learning Styles, and an Agile Approach to Software Engineering Education."ACM SIGCSE Bulletin, 38 (1) 2006.

McConnell, Jeffrey. J. "Active and Cooperative Learning: Further tips and tricks (part 3)." SIGCSE Bull 38(2)2006, 24-28.

Mishra, A., Cagiltay, Nerzie. E. and Kilic, O. "Software engineering education: some important dimensions." European Journal of Engineering Education, 32 (3) 2007, 349-361.

Petkovic, D., Thompson, G., and Todtenhoefer, R. "Teaching Practical Software Engineering and Global Software Engineering: Evaluation and Comparison."ACM SIGCSE Bulletin, 38 (3) 2006.

Richards Debbie."Designing Project-Based Courses with a Focus on Group Formation and Assessment."ACM Transactions on Computing Education, Vol. 9. No. 1, Article 2, March 2009.

Robillard, Pierre N. and Kruchten, Philippe. "Yoopeedoo (UPEDU): A process for teaching software process." Proceeding of the 14th Conference on Software Engineering Education and Training, Charlotte, North Carolina, Feb. 2001.

Sebern, Mark, "The Software Development Laboratory: Incorporating Industrial Practice in an Academic Environment."Proceedings of the 15th Conference on Software Engineering Education and Training (CSEET' 02), IEEE.2002.

The Joint Task Force on Computing Curricula, Software Engineering 2004: Curriculum Guidelines for Under-graduate Degree Programs in Software Engineering, IEEE Computer Society, Association of Computing Machinery, Aug. 2004.

Vliet, H. Van. Student Projects-Software Engineering: Principles and Practice (2nd Edition), Wiley, August 30, 2000.